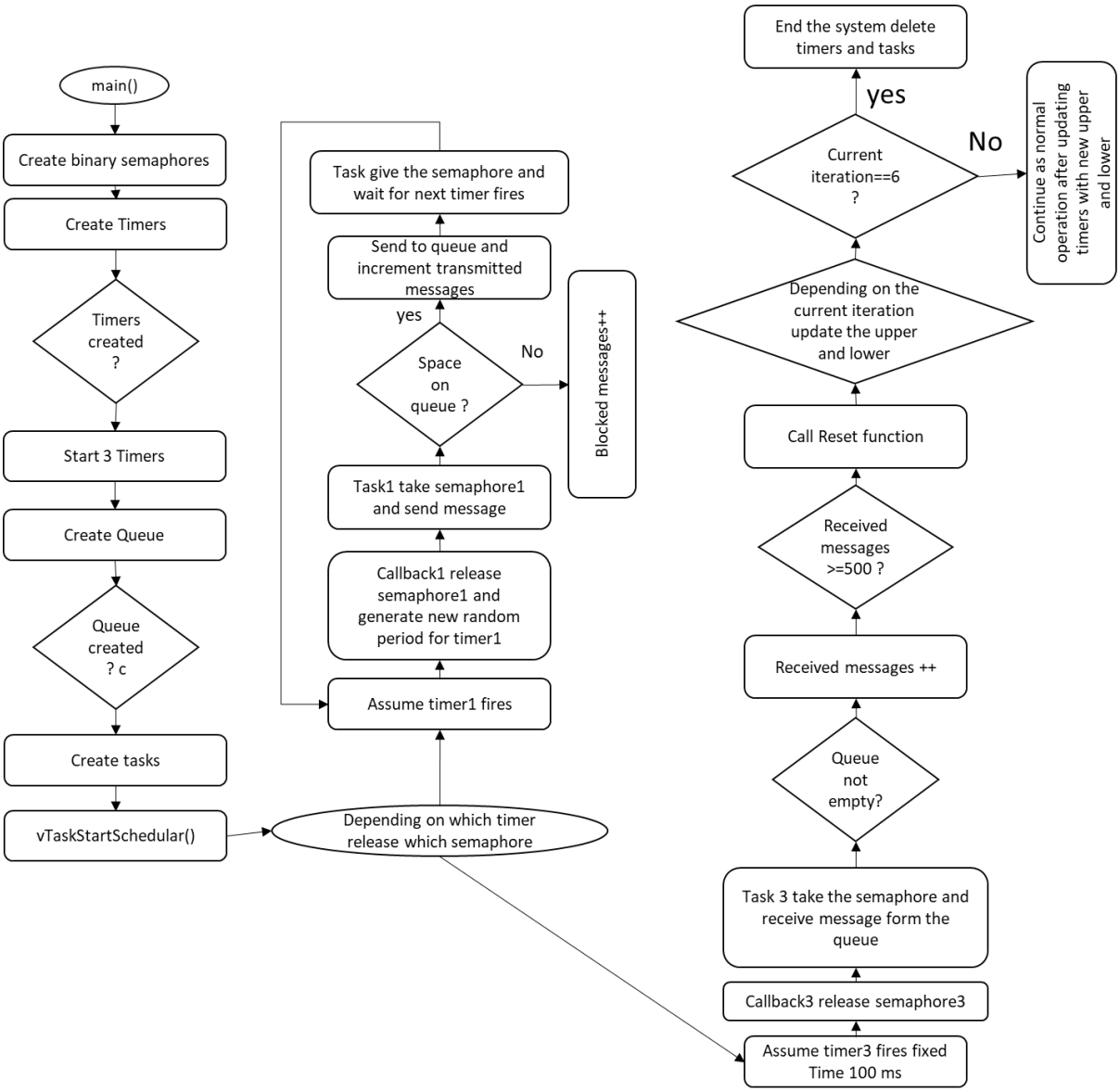
رقم مجموعة البلاك بورد	المقعد	الفصل	رقم الجلوس	الاسم
16	12	3	32122	فاروق خالد محمد محمد السيد
16	20	3	32130	

System Design section

- We have three tasks (2 senders and one receiver)three timers three binary semaphores the three 3 tasks synchronization through timers and communicate through queue of fixed size .

Over flow of the program explained in flow chart



Tasks synchronization

Each task blocked on a Binary semaphore which is released by the callback function of the related timer(3tasks , 3 timers , 3 semaphores)

(t_sender is random period depend on we are on which iteration) (upper and lower bound).

The change of period for timer happen on it's callback function .

Example Task1:

```
void vSenderTask1(void *pvParameters)
{
	BaseType_t xStatus;
	for (;; )
	{
		static TickType_t xTimeNow;
		xTimeNow = xTaskGetTickCount();
		xSemaphoreTake(xBinarySemaphore1, portMAX_DELAY);
//Task is waiting for binary semaphore which is released by the callback function of timer
		xStatus = xQueueSendToBack(xQueue, &xTimeNow, 0);

--------------- Task has more details on the code
```

The callback function of the timer1:

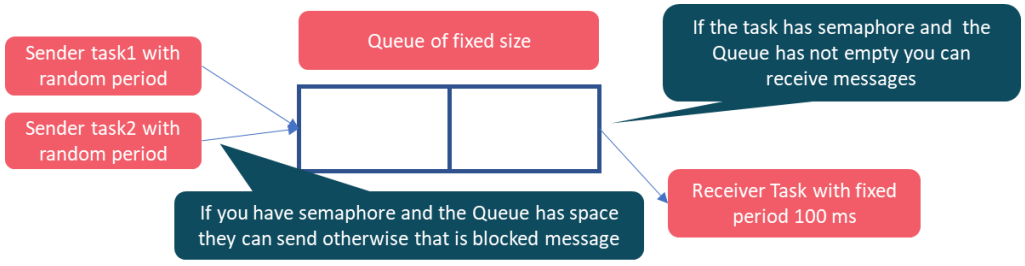
```
static void prvSender1TimerCallback(TimerHandle_t xTimer)
{
// when the timer fired it release the semaphore for task1
	xSemaphoreGive(xBinarySemaphore1);
	// the timer change the period with new other random period within the limit of the iteration(upper and lower)
	TickType_t new_Timer_period = randbetween (lower, upper);
	TickType_t new_Timer_period_in_Ticks = pdMS_TO_TICKS(new_Timer_period);
	xTimerChangePeriod(xTimer, new_Timer_period_in_Ticks, 0);

}
```

Similar logic for task2 (sender) and task3 (receiver) all wait for binary semaphore released by callback function of timer but tReceiver has fixed time 100ms .

How tasks communicate

Through queue has fixed size 2 sender Task send to it and only one receiver received from it .

- When any of the sender tasks have the semaphore and there is space on the queue it will send to the queue and increment the transmitted messages if there is no space it will increment the blocked messages .
- When Receiver task take the semaphore if there is messages on the queue it will read one message and sleep again for next fire



In the queue I send uint32 (TickType_t) then convert it to string in Receiver Task

```
void vReceiverTask3(void *pvParameters)
{
	TickType_t lReceivedValue;
	BaseType_t xStatus;
	for (;; )
	{
		xSemaphoreTake( xBinarySemaphore3, portMAX_DELAY );
		xStatus = xQueueReceive(xQueue, &lReceivedValue, 0);
		if (xStatus == pdPASS)
		  {
// we received data in queue as uint32 then converted it to string if we need print it.
			char str[80];
			 sprintf(str, "Time is = %d\r\n", lReceivedValue);
			 puts(str) ;
				total_number_of_recived_messages++ ;
// check if the number of received messages greater than or 500 call the reset function.
			}
			if (total_number_of_recived_messages>=500)
			{
				Reset();
			}
	}}
```

Reset function : Inside the reset function I do all the required on the project description .

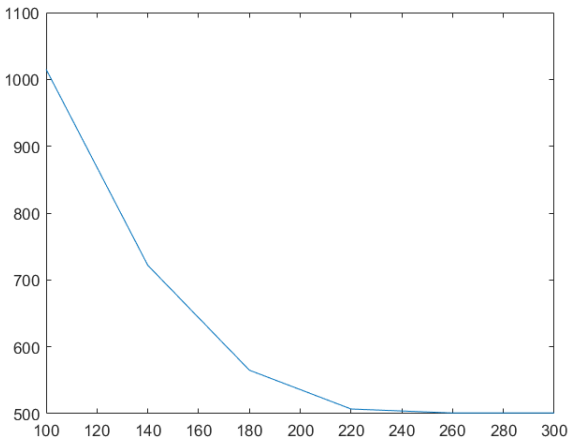
- As part of the design inside the reset function I increase the current iteration section which determine the upper and lower bound which used to get random period.
- I said before that I change the period of the timer to new timer period in its callback function.
- Current iteration 1 the bounds (80 ,120), if iteration 2 (110,250) if the current iteration is 6 I will print “Game Over delete all tasks and timers and end the system”

Results section : When the size of queue = 2

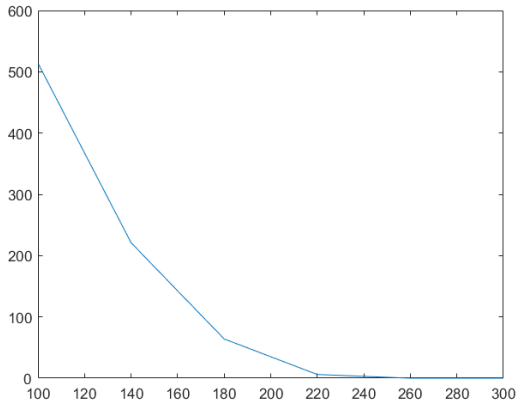
Tsender avg	100	140	180	220	260	300
Sent_messages	1015	722	565	507	501	501
Blocked_messages	514	221	64	6	0	0

The plot using matlab :

total sent messages as function of aveage sender :



number of blocked messages as function of average value of Tsener:



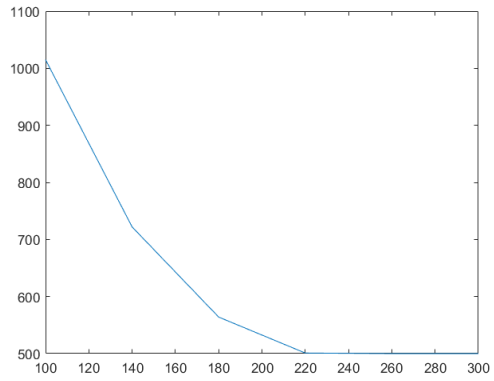
Explain the gap between the number of sent and received messages in the running period.

The gab is due to timing and queue size. The messages blocked if the queue was full. if time of the receiver faster than the time of the sender you have less blocked messages because receiver function will receive messages faster so let the space in queue not full so less blocked messages. that is why we you go to high lower and upper bound (ex 170, 350) the blocked messages zero because any random value will be greater than the receiver timer(100ms) so receiver function fast enough to let the queue empty .

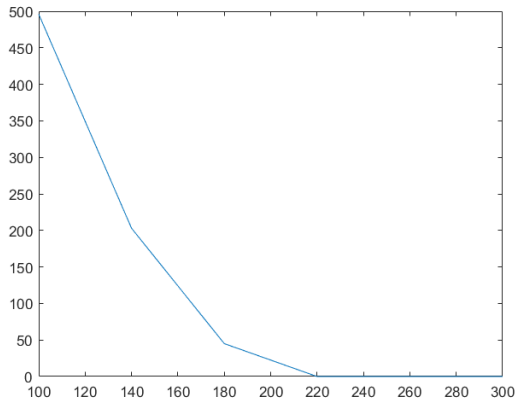
repeat for a queue of size 20

Tsender avg	100	140	180	220	260	300
Sent_messages	1015	722	564	501	500	500
Blocked_messages	496	203	45	0	0	0

total sent messages as function of aveage sender :



number of blocked messages as function of average value of Tsender:



What happens when queue size increases:

The number of blocked messages decreased . because there is more spaces to send in the queue not full . The receiver has more opportunity to make the queue not full . so increasing the size of the queue increase the probability to find queue not full so less blocked messages and at higher upper and lower bonds compare to the receiver time the blocked messages reaches zero .

References:

1-Mastering the FreeRTOS Real Time Kernel - a Hands On Tutorial Guide by Richard Barry

I read that reference and goes on its examples and take the naming convention from it an .